98647-000-73

Sartorius Data Input YDI 150

Gestionnaires d'interface
pour terminaux Data Input YDI 150

sartorius

Sommaire

	Page
Création d'une copie de travail de la disquette	78
Gestionnaire d'interface I/O numérique I2CDRV.BIN	79
Le gestionnaire des ports d'entrée/sortie (I/O)	79
Accès aux ports I/O	79
Le port de sortie numérique	80
Le port d'entrée numérique	83
Le port à interrupteurs DIP	84
Gestionnaire d'interface série «SER150»	87
Installation du gestionnaire SER150	88
Utilisation du gestionnaire SER150 dans un programme d'application	90
Commande OPEN pour le fichier gestionnaire	90
Commande WRITE (émission des données)	91
Commande READ (réception des données)	93
Commande WRITE pour le contrôle d'entrée et de sortie (I/O Control)	94
– Fonction 0: MOUNT (affecter la zone tampon d'entrée)	98
– Fonction 1: PURGE (effacer la mémoire tampon d'entrée)	98
– Fonction 2: SET PROTOCOL (définir les paramètres de protocole)	99
– Fonction 3: FLUSH INPUT (vider la mémoire tampon d'entrée)	103
– Fonction 4: FLUSH OUTPUT1 (vider la mémoire tampon de sortie 1)	103
– Fonction 5: FLUSH OUTPUT2 (vider la mémoire tampon de sortie 2)	103
– Fonction 6: FLUSH ALL (vider toutes les mémoires tampon)	103
– Fonction 7: SET UART-LINES (définir les lignes de commande UART)	104
– Fonction 8: WRITE BUFFER 2 (écrire dans la mémoire tampon de sortie 2)	105
– Fonction 9: Change UART (redéfinir UART)	106
– Fonction 10: Set MUX Channel (définir le canal mutiplexeur)	107
I/O Commande de contrôle READ	107
Programme V24_SET	110
Réglages SARTONET-MASTER	110
Réglages SARTONET-SLAVE	110
Autres réglages de protocole	111
Licence d'exploitation du logiciel Sartorius	112
Informations juridiques	115
Annexe: Fiche d'enregistrement SARTORIUS	
ci-joint: programme sur disquette de 3,5 pouces	

Création d'une copie de travail de la disquette

Nous vous conseillons vivement de constituer une copie de travail de votre disquette.

Si vous endommagez votre disquette de travail ou si vous effacez par inadvertance les fichiers qu'elle contient, vous aurez ainsi toujours les fichiers de la disquette originale à votre disposition.

La commande DISKCOPY copie le contenu d'une disquette sur une autre disquette. «DISKCOPY» est la façon la plus rapide qui soit de copier une disquette puisque cette commande permet de copier la totalité du contenu de la disquette en une seule opération.

Remarque!

Vous ne pouvez utiliser la commande DISKCOPY que si le lecteur cible et le lecteur source ont la même capacité.

Insérez la disquette dans le lecteur «A:» et entrez la commande suivante:

– sur un PC avec un seul lecteur de disquette:
`DISKCOPY`

– sur un PC avec deux lecteurs de disquette:
`DISKCOPY A: B:`

Appuyez ensuite sur la touche ENTER.

Si votre disquette cible n'est pas encore formatée, elle sera formatée pendant le processus de copiage.

Répondez ensuite aux messages sur l'écran et terminez le processus de copiage.

Conservez précieusement votre disquette originale!

Vous pouvez dès à présent travailler exclusivement avec les copies de travail. Conservez votre disquette originale dans un lieu sûr et utilisez celle-ci seulement pour la création de copies de travail supplémentaires.

Remplissez soigneusement votre fiche d'enregistrement et envoyez-la à votre revendeur Sartorius. Vous bénéficierez ainsi de notre service «Update» de mise à jour.

Gestionnaire d'interface I/O numérique I2CDRV.BIN

Les informations que suivent sont destinées aux programmeurs désirant développer des applications pour gérer les ports numérique d'entrée/sortie (I/O) sur le YDI 150. Les modèles de la série YDI 150 sont tous équipés de huit entrées numériques, quatre entrées à interrupteur DIP, huit sorties numériques et quatre sortie à relais avec un contact inverseur pour chacun d'eux (single-pole, double-through). Les ports sont connectés au niveau interne à un bus I^2C et sont accessibles par l'intermédiaire du gestionnaire de périphérique 12 CDVR.BIN.

Le gestionnaire des ports d'entrée/sortie (I/O)

Aux périphérique DOS correspondent des noms de fichiers réservés ayant chacun une signification particulière, par ex.: PRN est le périphérique d'impression par défaut et COM1 ou AUX sont les noms réservés au premier port série. Du point de vue du programmeur, un périphérique est traité comme un fichier et la gestion de ce fichier se fait par l'intermédiaire des fonctions d'entrée/sortie courantes, par ex.: READ, WRITE, OPEN et CLOSE, en utilisant le nom de fichier reçu.

Les ports numérique d'entrée et de sortie sur le YDI 150 utilisent le gestionnaire de périphérique 12 CDRV.BIN. Une fois que ce gestionnaire est installé, on utilise le nom de fichier réservé **«DIGI»** pour accéder aux ports d'entrée et de sortie. L'installation du gestionnaire se fait en entrant une commande DEVICE dans le fichier CONFIG.SYS avec un chemin d'accès au fichier 12 CDRV.BIN, par ex.:
DEVICE=D:\12CDRV.BIN

Remarque:
Si vous chaînez le contrôle de CONFIG.SYS à HDCONFIG.SYS, vous devez alors entrer l'instruction DEVICE dans le fichier HDCONFIG.SYS.

Accès aux ports d'entrée/sortie (I/O)

L'accès aux ports I/O se fait par la lecture/écriture de chaînes de contrôle en format ASCII au niveau du gestionnaire des ports après avoir ouvert le fichier de périphérique «DIGI» en mode binaire. Tout langage de programmation pouvant ouvrir, lire et écrire des fichiers DOS peut être utilisé, par ex.: C, Pascal, Basic…

Le premier caractère de la chaîne de contrôle indique le type du port auquel on veut accéder: entrée, sortie, interrupteur DIP ou relais. Les caractères suivants indiquent l'état des lignes d'entrée/sortie (I/O) individuelles. Les caractères 2, 3, 4… caractérisent les lignes 0, 1, 2… C'est donc le nombre de lignes individuelles I/O sur un port qui détermine la longueur de la chaîne. Les ports d'entrée et de sortie on huit lignes I/O; les ports à relais et à interrupteurs DIP en ont quatre.

Identificateurs de ports I/O:

Caractère ASCII **«O»** (0x4F) = Port de sortie (**O**utput) numérique
Caractère ASCII **«I»** (0x49) = Port d'entrée (**I**nput) numérique
Caractère ASCII **«R»** (0x52) = Port de sortie à **R**elais
Caractère ASCII **«D»** (0x49) = Port d'entrée à interrupteurs **D**IP

Etats de ligne:

Caractère ASCII **«0»** (0x30) = Logique 0
Caractère ASCII **«1»** (0x31) = Logique 1
Caractère ASCII **«X»** (0x58) = Inchangé

Exemple:
Pour activer les deux premiers relais et laisser les deux autres inchangés on doit écrire dans le fichier de périphérique DIGI la chaîne de contrôle suivante: **«R11XX»**.

Le port de sortie numérique

Le YDI 150 est équipé d'un port de sortie avec huit lignes de sortie TEC open-drain. Elle sont connectées au niveau interne à une tension de +5 volts par l'intermédiaire de résistances PULL-UP. Le réglage de ces lignes de sorties se fait en écrivant une chaîne de contrôle dans le fichier de périphérique «DIGI». La chaîne de contrôle est constituée de neuf caractères: le premier caractère «O» est l'identificateur du port de sortie et les huit caractères suivants représentent les lignes de sortie 0 à 7. Les caractères sont encodés comme suit:

Caractère ASCII **«0»** (0x30)→ TEC est conducteur; la sortie est tirée sur le potentiel de référence (0 volts)

Caractère ASCII **«1»** (0x31)→ TEC n'est pas conducteur; la sortie non-chargée passe à +5 volts

Caractère ASCII **«X»** (0x58)→ Inchangé. Pendant l'exécution de la fonction WRITE, le gestionnaire de périphérique transforme la chaîne de contrôle de façon à ce que le «X» soit remplacé par le réglage actuel de la sortie.

Exemple:
Vous désirez que le TEC soit conducteur pour la ligne de sortie 5 («Output line 5»), donc que cette ligne soit tirée sur le potentiel de référence. Les lignes de sortie sont initialement toutes non-conductrices, donc tirées sur 5 volts par l'intermédiaire des résistances PULL-UP. La chaîne de contrôle à écrire dans le gestionnaire de périphérique est alors la suivante: «OXXXXX0XX». Après l'écriture, la chaîne de contrôle se transfome en «O11111011»

Turbo-C:

```
int handle
char ctrl_str[10];
strcpy(ctrl_str,"0XXXXX0XX");
handle=open ("digi",O_WRONLY|O_BINARY);
write (handle,9,ctrl_str);
close(handle);
```

Turbo Pascal:

```
CONST  O_Binary = True;
VAR    Ctrl_Str   :Array[0..8] of Char;
       Handle     :File;

BEGIN
Ctrl_Str:="0XXXXX0XX";
Assign(Handle,´DIGI`);
ReSet(Handle,1);                    (N'ouvre pas de fichiers en
                                    mode binaire)
DeviceMode(Handle,O_Binary);        (Voir listing de la fonction DeviceMode
BlockWrite(Handle,Ctrl_Str,9);      en dernière page de cette première
Close(Handle);                      partie)
END.
```

Quick Basic:

```
OPEN "DIGI" FOR BINARY AS #2
CTRL_STR$="0XXXXX0XX"
PUT #2, , CTRL_STR$
CLOSE #2
```

Port de sortie à relais

Le YDI 150 est équipé d'un port à relais comprenant 4 relais (single-pole, double-through). Le réglage des relais se fait en écrivant une chaîne de contrôle dans le fichier de périphérique «DIGI». La chaîne de contrôle est constituée de 5 caractères: le premier caractère «R» est l'identificateur du port à relais et les 4 caractères suivants représentent les lignes de sortie 0 à 3. Les caractères sont encodés comme suit:

Caractère ASCII "0" (0x30)	→	Relais en position ON
Caractère ASCII "1" (0x31)	→	Relais en position OFF
Caractère ASCII "X" (0x58)	→	Inchangé.

Pendant l'exécution de la fonction WRITE, le gestionnaire de périphérique transforme la chaîne de façon à ce que le «X» soit remplacé par le réglage actuel de la sortie.

Exemple:
Vous désirez activer les relais 1 et 2. Tous les relais sont initialement en position OFF. La chaîne de contrôle à écrire dans le fichier de périphérique est alors la suivante: «RXX00X». Après l'écriture, la chaîne de contrôle est transformée en "R1001».

Turbo-C:
```
int handle
char ctrl_str[6];
strcpy(ctrl_str,"RXOOX");
handle=open ("digi",O_WRONLY|O_BINARY);
write (handle,5,ctrl_str);
close(handle);
```

Turbo Pascal:
```
CONST  O_Binary = True;
VAR    Ctrl_Str   :Array[O..4] of Char;
       Handle     :File;
```

```
BEGIN
Ctrl_Str:=´RXOOX`;
Assign(Handle,´DIGI`);
ReSet(Handle,1);                    (N'ouvre pas de fichiers en mode
                                    binaire)
DeviceMode(Handle,0_Binary);        (Voir listing de la fonction DeviceMode
BlockWrite(Handle,Ctrl_Str,5);      en dernière page de cette première
Close(Handle);                      partie)
END.
```

Quick Basic:

```
OPEN "DIGI" FOR BINARY AS #2
CTRL_STR$="RXOOX"
PUT #2, , CTRL_STR$
CLOSE #2
```

Le port d'entrée numérique

Le YDI 150 est équipé d'un port d'entrée avec huit lignes d'entrée TTL. Elle sont connectées au niveau interne à une tension de +5 volts par l'intermédiaire de résistances PULL-UP. Le réglage de ces lignes d'entrée est lu et mémorisé à partir du fichier de périphérique «DIGI» dans une chaîne de caractères comprenant 9 caractères. Le premier caractère identifie le port d'entrée et doit être initialisé avec «I» avant l'appel de la fonction read (lecture). Les huit autres caractères représentent les lignes d'entrée respectives 0 à 7 et sont retournés comme suit:

Caractère ASCII "0" (0x30)
La ligne d'entrée est court-circuitée sur le potentiel de référence, ou une tension de moins de 0,8 volts lui est appliquée par rapport au potentiel de référence.

Caractère ASCII "1" (0x31):
Une tension de plus de 0,8 vots est appliquée à la ligne d'entrée par rapport au potentiel de référence ou la ligne d'entrée n'est pas connectée et donc réglée sur +5 volts par l'intermédiaire des résistances PULL-UP internes.

Exemple:
Les entrée 3 et 4 sont court-circuitées sur le potentiel de référence. Les autres entrées ne sont pas connectées. Après la lecture, la chaîne de caractères est transformée en «I11100011».

Turbo-C:

```
int handle
char ctrl_str[10];
ctrl_str[0]="I";
handle=open ("digi",O_RDONLY|O_BINARY);
read (handle,9,ctrl_str);
close(handle);
```

Turbo Pascal:

```
CONST  O_Binary = True;
VAR    Ctrl_Str   :Array[0..8] of Char;
       Handle     :File;

BEGIN
Ctrl_Str[0]:="I";
Assign(Handle,´DIGI`);
ReSet(Handle,1);                    (N'ouvre pas de fichiers en mode
                                    binaire)
DeviceMode(Handle,O_Binary);        (Voir listing de la fonction DeviceMode
BlockRead(Handle,Ctrl_Str,9);       en dernière page de cette première
Close(Handle);                      partie)
END.
```

Quick Basic:

```
CTRL_STR$="IXXXXXXXX":          REM pour prédéfinir la longueur
OPEN "DIGI" FOR BINARY AS #2    de la chaîne
GET #2, 1, CTRL_STR$:           REM pour lire à partir de la position 1
CLOSE #2                        dans le fichier
```

Le port à interrupteurs DIP

Le YDI 150 est équipé d'un port d'entrée contrôlé par 4 interrupteurs DIP situés au niveau du socle du YDI 150. Le réglage des interrupteurs DIP est lu et mémorisé à partir du fichier de périphérique «DIGI» dans une chaîne de contrôle constituée de 5 caractères. Le premier caractère identifie le port à interrupteurs DIP et doit être initialisé avec «D» avant l'appel de la fonction read (lecture). Les 4 autres caractères représentent les interrupteurs DIP respectifs 0 à 3 et sont retournés comme suit:

Caractère ASCII "0" (0x30) → L'interrupteur DIP est relié à la masse.

Caractère ASCII "1" (0x31) → L'interrupteur DIP reste ouvert.

Exemple:
Les interrupteurs 0 et 1 sont fermés et les interrupteurs 2 et 3 sont ouverts. Après la lecture, la chaîne de contrôle est transformée en «D0011».

Turbo-C:

```
int handle
char ctrl_str[6];
ctrl_str[0]='D`;
handle=open ("digi",O_RDONLY|O_BINARY);
read (handle,5,ctrl_str);
close(handle);
```

Turbo Pascal:

```
CONST  O_Binary = True;
VAR    Ctrl_Str   :Array[0..4] of Char;
       Handle     :File;

BEGIN
Ctrl_Str[0]:='D`;
Assign(Handle,'DIGI`);
ReSet(Handle,1);                    (N'ouvre pas die fichiers en mode
                                    binaire)
DeviceMode(Handle,O_Binary);        (Voir listing de la fonction DeviceMode
BlockRead(Handle,Ctrl_Str,4);       en dernière page de cette première
Close(Handle);                      partie)
END.
```

Quick Basic:

```
CTRL_STR$="DXXXX":                  REM pour prédéfinir la longueur
OPEN "DIGI" FOR BINARY AS #2        de la chaîne
GET #2, 1, CTRL_STR$:               REM pour lire à partir de la position 1
CLOSE #2                            dans le fichier
```

La fonction «DeviceMode» en Turbo Pascal

Lorsqu'on ouvre un périphérique de caractères DOS avec les fonctions d'ouverture de Turbo PASCAL courantes (Reset, ReWrite ou Append), il n'est pas possible d'omettre le filtrage d'entrée/sortie en réglant le périphérique en mode binaire. Pour y remédier, on peut appeler DeviceMode après avoir ouvert le périphérique. La fonction DOS $44 est ainsi appelée (IOCTL) et le fichier transformé en «Handle» est réglé en mode binaire si «BinMode» est une variable «vrai». DeviceMode retourne TRUE si le changement de mode a été exécuté correctement.

```
Function DeviceMode(Var Handle: File; BinMode: Boolean):
Boolean;
VAR Regs: Registers;          (Utilise l'unité DOS)
BEGIN
With Regs DO
 BEGIN
 AX:= $4400;
 BX:= FileRec(Handle).Handle;
 IF BX<>0
  THEN BEGIN                  (Continuer si file handle est valide)
  MsDos(Regs);                (Lire les réglages actuels du périphérique)
  IF (DX AND $0080)<>0
   THEN BEGIN                 (Ne changer le mode que si handle est un
                               périphérique de caractères DOS)
   AX:= $4401;
   DH:= 0;
   IF BinMode
    THEN DL:= DL OR $20       (Régler le mode binaire)
    ELSE DL:= DL AND $DF;     (Régler le mode de traduction de caractère)
   MsDos(Regs);
   DeviceMode:= True;
   END
   ELSE DeviceMode:= False;
  END
  ELSE DeviceMode:= False;
 END;                         (Quitter DOS)
END;                          (Quitter DeviceMode)
```

Gestionnaire d'interface série «SER150»

Le pilote d'interface «SER150» gère l'entrée et la sortie des données de transmission d'une interface série. Ce gestionnaire est intégré dans le système d'exploitation (OS) en tant que gestionnaire OS de périphérique (OS DEVICE DRIVER). De plus, vous pouvez utiliser ce gestionnaire avec les commandes standard d'entrée et de sortie du compilateur de votre choix, sans avoir besoin de programme d'interface DOS-Application-Program-Interface (DOS-API).
Le gestionnaire «SER150» offre les possibilités fonctionnelles suivantes:

Divers modes de protocole (orientation des blocs et des caractères):
– transmission continue caractère par caractère (free running)
– fin de bloc définie par un caractère spécial
– fin de bloc définie par l'entrée de la longueur du bloc
– fin de bloc définie par le dépassement du temps prévu (timeout) entre les caractères
– protocole SARTONET Master/Slave (avec matériel d'interface spécial)

Contrôle du dépassement du temps prévu (Timeout check):
3 valeurs de timeout programmables sont disponibles:
– lors de l'écriture (WRITE Timeout)
– lors de la lecture (READ Timeout)
– lors de l'attente du démarrage d'un bloc (Block Start Timeout)

Paramètres de transmission programmables
– vitesse de transmission (110 jusqu'à 38400 bauds)
– longueur de mot
– parité
– bits de stop

Contrôle individuel et automatique et activation des signaux de colloque («handshake lines»):
– DTR (Data Terminal Ready)
– RTS (Request To Send)
– DSR (Data Set Ready)
– RI (Ring Indicator)
– CTS (Clear To Send)
– DCD (Data Carrier Detect)
– XON / XOFF protocol

La taille des mémoires tampon d'entrée et de sortie est programmable.

Installation du gestionnaire «SER150«

Pour installer le gestionnaire de périphérique SER150 sur un ordinateur MS-DOS, il faut entrer une des lignes d'instructions suivantes dans le fichier CONFIG.SYS qui se trouve dans le répertoire racine (Root Directory) du lecteur BOOT (ce fichier est habituellement C:\CONFIG.SYS):

```
DEVICE=<path>\SER150.SYS <buffsize>
<port> <interrupt> <name>
```

ou `DEVICE=<path>\SER150.SYS <name>`

ou `DEVICE=<path>\SER150.SYS`

Les paramètres ont la signification suivante:

<path>
Chemin de recherche complet du répertoire qui contient le fichier SER150.SYS.

<buffsize>
Longueur de la zone tampon d'entrée en paragraphes de 16 octets sous la forme d'une valeur hexadécimale, par exemple:

800 = 8000 octets hex. = 32 K-octets
réglage par défaut: 80 = 800 octets hex. = 2 K-octets

<port>
Adresse I/O du port d'interface série que vous voulez utiliser, sous la forme d'une valeur hexadécimale:
pour COM1, entrer 3F8,
pour COM2, entrer 2F8
et pour MUX entrer 300
Réglage par défaut: 3F8 (COM1)

<interrupt>
Numéro de la la demande d'interruption du port d'interface série, sous la forme d'une valeur hexadécimale:
pour COM1, entrer 0C
pour COM2, entrer 0B
et pour MUX, entrer 0F
Réglage par défaut: 0C (COM1)

<name>
Nom d'accès au fichier gestionnaire de périphériques (Device-Driver). Ce nom doit comporter au maximum 8 lettres/caractères.

Remarque!
Assurez-vous qu'aucun autre fichier ou répertoire ne possède le même nom (l'extension <name>.<ext> ne joue aucun rôle dans la différenciation), sinon vous perdez définitivement l'accès à ce fichier ou à ce répertoire!
par exemple: V24.
Réglage d'usine: V24SART.

Vous pouvez omettre les paramètres <buffsize>, <port>, <interrupt> et <name>. C'est la valeur par défaut qui sera dans ce cas utilisée.
Exemple:

```
CONFIG.SYS:
FILES=20
BUFFERS=20
DEVICE=C:\DRIVERS\SER150.SYS 200 2F8 0B SER2
```

Le fichier CONFIG.SYS installe le gestionnaire SER150.SYS qui se trouve dans le répertoire C:\DRIVERS. Une mémoire tampon d'entrée de 2000 hex (= 8 K-octets) est réservée. Le port série sélectionné est COM2 et le nom pour la commande OPEN est «SER2».

Utilisation du gestionnaire «SER150» dans un programme d'application

Vous pouvez utiliser le gestionnaire avec la commande INT 21h appropriée du système d'exploitation MS-DOS si vous programmez en Assembler. Vous pouvez aussi utiliser les commandes I/O d'un langage de plus haut niveau tel que par exemple «C» ou «Pascal». L'exemple d'application que nous vous proposons ci-dessous est conçu pour un compilateur «C». Si vous travaillez avec un autre compilateur, veuillez vous référer pour plus de détails au manuel correspondant ou à la documentation du système d'exploitation DOS.

Commande OPEN pour ouvrir le fichier gestionnaire

Avant d'envoyer des caractères, vous devez ouvrir le fichier gestionnaire avec la fonction OPEN, par exemple:

```
void main(void)
    {
    int  v24_handle;
    v24_handle = open("ser1",O_RDWR |
    O_BINARY);
                 /* open for READ and WRITE in
                 BINARY mode */
    if (v24_handle == -1)
       {
       printf( Error at OPEN of device ser1\n );
       exit(9);/* error-exit */
       }
    else
       /* Ok, no error during open */
       .
       .
       .
    }
```

Dans le langage de programmation «C», vous devez mémoriser pour «handle» une valeur numérique de 2 octets qui est retournée par la fonction OPEN étant donné que toutes les fonctions I/O se réfèrent à cette valeur.

Commande WRITE (émission des données)

Vous pouvez utiliser la fonction WRITE pour envoyer des données via le port d'interface série. La chaîne de caractères reste inchangée lors de l'envoi (c'est-à-dire qu'aucun caractère de fin de bloc ne vient s'y ajouter). Seulement dans le cas d'un protocole SARTONET viennent s'ajouter à la chaîne de caractères les caractères d'en-tête de protocole et de conclusion de protocole nécessaires. Si vous avez activé le contrôle des lignes de modem, le gestionnaire SER150 vérifie ces lignes avant chaque transmission de caractères et contrôle également les dépassements du temps prévu. Si le temps prévu est dépassé, la transmission est interrompue. C'est pourquoi nous vous recommandons de vérifier le nombre des caractères transmis qui sont retournés par la fonction WRITE.

Voici l'exemple d'une fonction qui émet une chaîne de caractères et contrôle le résultat:

```
int writestring(int v24_handle, char *output)
    {
    int len = strlen(output);
    if ( write(v24_handle, output, len) == len )
       return OK;
    else
       return NOT_OK;
    }
```

Si vous avez sélectionné un protocole «SARTONET-Master», le premier octet des données de sortie n'est pas transféré puisqu'il définit l'adresse du «SARTONET-Slave» auquel les données sont destinées. Si la commande WRITE retourne le paramètre «0», cela signifie que l'esclave n'était pas opérationnel. Si le retour est «1 caractère émis», l'esclave a certes répondu, mais il n'a pas été en mesure de réceptioner les données (tampon d'entrée encombré!).

Exemple:

```
int writestring_master(int v24_handle, int
adress, char *output)
     {
     char buffer[300];
     int len = strlen(output);
     int status;

     buffer[0] = adress;
     strcpy(buffer+1, output);

     status = write(v24_handle, buffer, len+1)
     if (status == 0)
        printf("Slave not connected\n");

     if (status == 1)
        printf("Slave is busy\n");

     if (status == len+1 )
        return OK;
     else
        return NOT_OK;
     }
```

Commande READ (réception des données)

Vous pouvez utiliser la fonction READ pour recevoir des caractères depuis le port d'interface série. Lors du tranfert, le bloc de caractères reste inchangé et le caractère de fin de bloc est inclus. Avec la commande READ, le nombre maximal de caractères devant être lus est certes prédéfini, mais le gestionnaire SER150 n'émet que les caractères jusqu'à la fin du transfert de bloc en cours. Si la fonction READ lit un nombre de caractères inférieur à celui compris dans le bloc, les caractères restants seront mémorisés et pourront être lus avec la commande READ suivante. Le gestionnaire SER150 peut être programmé pour reconnaître la fin d'un bloc par:

– un caractère de fin de bloc
– un dépassement du temps prévu entre deux caractères
– une longueur constante donnée
– le protocole SARTONET

La fonction suivante lit par exemple une chaîne de caractères d'une longueur maximale prédéfinie qui se termine par le caractère ASCII «NUL»:

```
int readstring(int v24_handle, char *input, int maxlen)
    {
    int len;
    len = read(v24_handle, input, maxlen);

    if (len == -1)
       {
       printf("Read error at device SER1\n");
       return NOT_OK;
       }

    if (len == 0)
       {
       printf("No data at device SER1\n");
       return NOT_OK;
       }

    input[len] = 0; /* terminate string (for C) */
    return OK;
    }
```

Si vous avez sélectionné le protocole «SARTONET-Master», Le premier octet de la chaîne de caractères à entrer doit déjà comporter l'adresse «SARTONET-Slave» avant l'appel de la fonction READ. La fonction READ retourne «0 caractères lus» si le «SARTONET-Slave» ne répond pas en l'espace de 50 milli-secondes. Elle retourne «1 caractère lu» si le «SARTONET-Slave» répond mais qu'il n'a aucune donnée disponible. Une fois que la lecture avec la fonction READ a été exécutée correctement, le premier octet de la mémoire tampon d'entrée contient l'adresse «SARTONET-Slave».

Commande WRITE pour le contrôle d'entrée/sortie (I/O-Control)

La commande WRITE de contrôle I/O (IOCTL) sert à définir les paramètres caractéristiques et les paramètres de transmission. Les réglages que vous effectuez restent toujours valides même après la fermeture du fichier gestionnaire de périphérique. Etant donné que la commande IOCTL-WRITE n'est pas supportée par les pilotes de périphériques courants DOS, il se pourrait qu'il existe également des compilateurs qui ne disposent pas non plus de cette fonction.

Le compilateur doit pouvoir permettre utilisation de l'interruption 21h, fonction 44h/sous-fonction 03h. Il faut sinon utiliser le programme V24_SET pour régler les paramètres du gestionnaire (cf. section «le programme V24_SET»). La description des fonctions C se réfère à la structure de données suivante:

```
#define IOCTL_READ        2
#define IOCTL_WRITE       3
#define IOCTL_INSTAT      6
#define IOCTL_OUTSTAT     7
#define STOPBIT_1         0x0000
#define STOPBIT_2         0x0001
#define WORDLEN_5         0x0000
#define WORDLEN_6         0x0002
#define WORDLEN_7         0x0004
#define WORDLEN_8         0x0006
#define PAR_EVEN          0x0000
#define PAR_ODD           0x0008
#define PAR_ZERO          0x0010
#define PAR_NONE          0x0018
#define BAUD_110          0x0000
#define BAUD_150          0x0020
#define BAUD_300          0x0040
#define BAUD_600          0x0060
#define BAUD_1200         0x0080
#define BAUD_2400         0x00a0
#define BAUD_4800         0x00c0
#define BAUD_9600         0x00e0
#define BAUD_19200        0x0100
#define BAUD_38400        0x0120
#define CONTROL_RTS       0x0400
#define CONTROL_DTR       0x0800
#define CHECK_XONXOFF     0x0200
#define CHECK_CTS         0x1000
#define CHECK_DSR         0x2000
#define CHECK_RI          0x4000
#define CHECK_CD          0x8000
```

```
#define MODE_FREERUN     0
#define MODE_ENDCHAR     1
#define MODE_BLOCKLEN    2
#define MODE_TIMEOUT     3
#define MODE_SNETSLAVE   4
#define MODE_SNETMASTER  5

#define ERR_OVERRUN      0x0001
#define ERR_PARITY       0x0002
#define ERR_FRAMING      0x0004
#define ERR_OVERFLOW     0x0008
#define ERR_SENDTIME     0x0010
#define ERR_RECTIME      0x0020
#define ERR_WAITTIME     0x0040
#define ERR_BLKCHECK     0x0080
#define STAT_OUTBUFF2    0x0100
#define STAT_OUTBUFF1    0x0200
#define STAT_INBLOCK     0x0400
#define STAT_INBUFFER    0x0800
#define STAT_CTS         0x1000
#define STAT_DSR         0x2000
#define STAT_RI          0x4000
#define STAT_DCD         0x8000
#define MOUNT            0
#define PURGE            1
#define SET_PROTOCOL     2
#define FLUSH_INPUT      3
#define FLUSH_OUTPUT1    4
#define FLUSH_OUTPUT2    5
#define FLUSH_ALL        6
#define SET_UARTLINE     7
#define WRITE_BUFFER2    8
#define CHANGE_UART      9
```

```
struct      io_control
      {
            unsigned int      dev_status;
            unsigned int      dev_len;
            unsigned int      dev_bgn;
            unsigned int      dev_prm;
            unsigned char     protmode;
            unsigned char     protmode_2;
            unsigned int      endvalue;
            unsigned int      obuff_size;
            unsigned int      uartset;
            unsigned long     timea;
            unsigned long     timeb;
            unsigned long     timec;
            unsigned char     xon_char;
            unsigned char     xoff_char;
      }     io_ctrl;

struct s_setline {
            unsigned int      func;
            unsigned int      rts;
            unsigned int      dtr;
            unsigned int      tx_break;
      }     setline;

struct s_write_2nd_buff {
            unsigned int      func;
            char                    data[20];
      } buff2;

typedef struct s_change_uart {
            unsigned int      func;
            unsigned int      port;
            unsigned char     int_number;
      }     t_change_uart;
int   fh;
```

```
void main(void)
    {
    int  end;
    int  error;
    int  status;
    char buffer[100];

    fh = open("V24",O_RDWR | O_BINARY);
    io_ctrl.dev_status = SET_PROTOCOL;
    io_ctrl.timea = 50000L;      /* 50 milli-
                                    second */
    io_ctrl.timeb = 1000000L;    /* 1  second   */
    io_ctrl.timec = 10000000L;   /* 10 seconds */
    io_ctrl.uartset =
         BAUD_9600 | STOPBIT_1 | WORDLEN_7 |
         PAR_EVEN;
    io_ctrl.protmode= MODE_ENDCHAR;
    io_ctrl.endvalue= 10;        /* Block
                                    terminator =
                                    LineFeed */
    status = ioctl(fh,IOCTL_WRITE,&io_ctrl,
    sizeof(io_ctrl));
    printf( \nIOCTL returns: %d\n ,status);
    close(fh);
    }
```

L'exemple de programme ci-dessus est un exemple de réglage des paramètres de protocole pour le gestionnaire SER150. Toutes les commandes IOCTL-WRITE disponibles sont détaillées dans les pages suivantes. La fonction requise est définie dans le premier champ de nombre entier «dev_status» de 2 octets du format de langage C «io_ctrl».

Fonction 0: MOUNT (affecter la zone tampon d'entrée)

Avec le système MS-DOS, la taille et l'emplacement de la mémoire tampon d'entrée est définie dans le fichier «CONFIG.SYS» pendant le processus d'amorçage du système. Avec la fonction 0, vous pouvez aussi déplacer la mémoire tampon d'entrée. Pour cela, la fonction nécessite l'information suivante dans le format de langage C «io_ctrl»:

`dev_status`	2-Byte Integer (nombre entier de 2 octets) 0 définit la fonction 0 (mount)
`dev_len`	2-Byte Integer taille du tampon (buffer) en paragraphes de 16 octets
`dev_bgn`	2-Byte Integer Segment d'amorçage du tampon en paragraphes de 16 octets
`dev_prm`	2-Byte Integer taille du tampon d'entrée utilisée. Ce champ ne revêt une importance que si vous lisez l'information IOCTL avec la commande IOCTL-READ. Excepté dans le cas de l'utilisation de «SARTONET», cette valeur est égale à la longueur du tampon en paragraphes (dev_len). Avec «SARTONET», cette valeur est diminuée par la taille de l'emplacement de mémoire pour le tampon de sortie.

La fonction 0 ne reconnait pas les autres paramètres.

Fonction 1: PURGE (effacer la mémoire tampon d'entrée)

Vous pouvez également utiliser cette commande dans l'environnement DOS, mais la mémoire tampon n'est pas désaffectée par cette fonction. La désaffectation de la zone de mémoire doit se faire par l'intermédiaire du programme. Les seules commandes auxquelles vous pouvez accéder sans qu'une mémoire tampon ne soit affectée sont OPEN, CLOSE, IOCTL WRITE et IOCTL READ.

`dev_status`	2-Byte Integer 1 définit la fonction 1 (purge)

La fonction 1 ne reconnait pas les autres paramètres.

Fonction 2: SET PROTOCOL (définir les paramètres de protocole)

Cette fonction est utilisée par le programme «V24_SET» et peut également être appelée à partir de votre programme d'application pour fixer les paramètres d'exploitation du gestionnaire SER150. Vous pouvez définir la vitesse de transmission, le format des caractères, les caractères de fin de bloc et les valeurs pour le timeout (dépassement du temps prévu).
Les paramètres sont:

dev_status 2-Byte Integer
2 définit la fonction 2 (set_protocol)

dev_len 2-Byte Integer
non-applicable pour la fonction 2

dev_bgn 2-Byte Integer
non-applicable pour la fonction 2

dev_prm 2-Byte Integer
non-applicable pour la fonction 2

protmode 1-Byte Integer
Définit le mode protocole pour le gestionnaire:
0 régime continu. Pas de terminaison de blocs.
1 caractère de fin de bloc. Les blocs sont terminés par un caractère de fin de bloc (cf. «endvalue»).
2 longueur de bloc. Les bloc ont une longueur constante (cf. «endvalue»).
3 Dépassement du temps prévu. Les blocs sont terminés avec comme valeur de timeout «timeb» après le dernier caractère.
4 SARTONET-Slave. Le gestionnaire se comporte comme un esclave SARTONET.
5 SARTONET-Master. Le gestionnaire se comporte comme un maître SARTONET.

protmode_2 1-Byte Integer
Adresse SARTONET pour l'esclave SARTONET.

endvalue 2-Byte integer
Pour le mode protocole 1: valeur ASCII du caractère de terminaison du bloc
Pour le mode protocole 2: longueur constante des blocs d'arrivée
Pour le mode protocole 4: longueur du tampon de sortie 2 en octets.

obuff_size 2-Byte integer
Longueur du tampon de sortie en octets. Si la longueur n'est pas «0», la commande WRITE n'attend pas jusqu'à-ce que des caractères soient écrits, elle retourne immédiatement un paramètre. Pour le mode protocole-SARTONET, une zone tampon doit être définie. En mode SARTONET, «obuff_size» fixe également la longueur maximale des informations d'entrées.

Remarque!

Assurez-vous qu'un emplacement de mémoire suffisant soit réservé (MOUNTed) aux tampons, de façon à ce que tous les tampons d'entrée et de sortie puissent être contenus. Cette valeur doit être calculée de la façon suivante:
– Mémoire tampon nécessaire [bytes]= 2 * obuff_size + endvalue + 16 (mode SARTONET)
ou
– Mémoire tampon nécessaire [bytes] = obuff_size + 16 (pour tous les autres protocoles)
Si l'emplacement de mémoire n'est pas suffisant pour les tampons, la commande IOCTL WRITE retourne la valeur pour «0 caractères transmis».

uartset 2-Byte integer

Bit 0:	bits de stop	
	0	= 1 bit de stop
	1	= 2 bits de stop
Bit 1-2:	longueur du mot	
	00	= 5 bits
	01	= 6 bits
	10	= 7 bits
	11	= 8 bits
Bit 3-4:	Parité	
	00	= pair
	01	= impair
	10	= zéro
	11	= pas de parité

Bit 5-8:	vitesse de transmission (110..38400) *	
	0000	= 110 bits/s
	0001	= 150 bits/s
	0010	= 300 bits/s
	0011	= 600 bits/s
	0100	= 1200 bits/s
	0101	= 2400 bits/s
	0110	= 4800 bits/s
	0111	= 9600 bits/s
	1000	= 19200 bits/s
	1001	= 38400 bits/s
Bit 9:	Protocole XON/XOFF	
	0	= inactif
	1	= actif XON / XOFF
Bit 10:	Commande RTS	
	0	= pas de commande RTS
	1	= la ligne de commande RTS est activée avant l'émission et elle est ensuite remise à zéro.
Bit 11:	Commande DTR	
	0	= pas de commande DTR
	1	= la ligne de commande est activée lorsque plus de 16 octets sont disponibles dans le tampon d'entrée et elle est remise à zéro s'il y a moins de 16 octets libres dans le tampon d'entrée.
Bit 12:	Contrôle CTS	
	0	= pas de contrôle de la ligne CTS
	1	= contrôle de la ligne CTS avant l'émission
Bit 13:	Contrôle DSR	
	0	= pas de contrôle DSR
	1	= contrôle de la ligne DSR avant l'émission

* = Le software du driver V24_SART permet une vitesse de transmission allant jusqu'à 38400 bits/s. Tous les matériels ne sont pas forcément suffisament rapides pour recevoir correctement les données à toutes les vitesses sans qu'il intervienne pendant la réception des erreurs dues à un dépassement de capacité.

Bit 14:	Contrôle RI	
	0	= pas de contrôle de la ligne RI
	1	= vérification de la ligneRI avant l'émission
Bit 15:	Contrôle CD	
	0	= pas de contrôle de la ligne CD
	1	= contrôle de la ligne CD avant l'émission

timea 4-Byte long integer
Timeout lors de l'envoi de données en micro-secondes (résolution de l'horloge = 1 milli-seconde)

timeb 4-Byte long integer
Timeout lors de la réception du bloc du début du bloc (premier caractère) jusqu'à la fin du bloc. En mode protocole 3 (timeout), cette valeur définit le dépassement de temps entre deux caractère qui a pour effet de terminer le bloc-transfert.

timec 4-Byte long integer
Timeout pendant une commande READ (depuis le début de la commande READ jusqu'à l'arrivée du premier caractère du bloc)

xon_char 1-Byte integer
Code ASCII du caractère qui sert de caractère XON pour le protocole XON/XOFF

xoff_char 1-Byte integer
Code ASCII du caractère qui sert de caractère XFF pour le protocole XON/XOFF

Remarque!

Paramètres de transmission SARTONET
Par principe, il est possible de combiner les modes protocole-SARTONET 4 et 5 avec n'importe lequel des paramètres de transmission qui sont disponibles. Notez cependant que le gestionnaire ne travaille bien sûr qu'avec d'autres appareils SARTONET dont la configuration est la suivante:

```
io_ctrl.timea = 50000L;   /* 50 millisecond */
io_ctrl.timeb = 1000000L; /* 1  second   */
io_ctrl.uartset =
      BAUD_9600 | STOPBIT_1 | WORDLEN_7 |
      PAR_EVEN;
```

Fonction 3: FLUSH INPUT (vider la mémoire tampon d'entrée)

La fonction 3 sert à vider (flush) la mémoire tampon d'entrée. Cette fonction efface tous les blocs que contient la mémoire tampon d'entrée.

Fonction 4: FLUSH OUTPUT1 (vider la mémoire tampon de sortie 1)

La fonction 4 efface le contenu de la mémoire tampon de sortie 1. Cette fonction ne peut être utilisée que si la mémoire tampon de sortie a été définie.

En mode protocole-SARTONET, cette fonction ne peut pas être exécutée si le transfert du contenu de la mémoire tampon a déjà commencé.

Fonction 5: FLUSH OUTPUT2 (vider la mémoire tampon de sortie 2)

La fonction 5 efface le contenu de la mémoire tampon de sortie 2. Cette mémoire tampon n'est disponible qu'en mode SARTONET. La fonction ne peut pas être exécutée si le transfert du contenu de la mémoire tampon a déjà commencé.

Fonction 6: FLUSH ALL (vider toutes les mémoires tampon)

La fonction FLUSH ALL est une combinaison des fonctions 3, 4, 5 et 6. Elle permet d'effacer toutes les données d'entrée et de sortie.

Fonction 7: SET UART-LINES (régler les lignes de commande UART)

Avec la fonction *7*, vous pouvez régler manuellement les lignes de commande UART. Si vous avez choisi le contrôle DTR ou le contrôle RTS dans la variable «uartset» avec la fonction SET PROTOCOL-2, la valeur que vous avez saisie avec la fonction SET UART-LINES sera surinscrite lors de la commande READ ou WRITE suivante.

Remarque!

Si la ligne de commande DTR a été réglée manuellement et si le gestionnaire de périphérique est programmé en mode SARTONET, cela provoque dans le meilleur des cas un mauvais fonctionnement du protocole SARTONET mais peut aussi détruire les chips des pilotes de lignes du matériel.

La fonction *7* utilise un format de langage C différent:

```
struct s_setline {
            unsigned int    func;
            unsigned int    rts;
            unsigned int    dtr;
            unsigned int    tx_break;
    }       setline;

void main(void)
    {
    fh = open( V24 ,O_RDWR | O_BINARY);
          ....
    setline.func =          SET_UARTLINE;
    setline.rts =           1;
    setline.dtr =           1;
    setline.tx_break =      1;
    ioctl(fh,IOCTL_WRITE,&setline,
    sizeof(setline));
          ....
    }
```

Les champs de «setline» ont la signification suivante:

func 2-byte integer (nombre premier de 2 octets)
7 définit la fonction *7* (SET UART-LINES).

rts 2-Byte integer
0 règle «Request To Send (RTS)» sur 0
1 règle RTS sur 1
2 laisse RTS inchangé

`dtr`	2-Byte integer 0 règle «Data Terminal Ready (DTR)» sur 0 1 règle DTR sur 1 2 laisse DTR inchangé
`tx_break`	2-Byte integer 0 règle «Transmit line (Tx)» sur 0 (condition d'interruption) 1 règle Tx sur 1 (réglage standard sans transmission) 2 laisse Tx inchangé

Fonction 8:
WRITE BUFFER 2 (entrer des données dans la mémoire tampon de sortie 2)

Cette fonction est utilisée dans le mode SARTONET-Slave pour écrire des données dans la mémoire tampon de sortie 2. Cette mémoire tampon a une priorité supérieure à la mémoire tampon de sortie 1 dans laquelle sont inscrites les données par l'intermédiaire de la fonction WRITE standard. La commande WRITE BUFFER2 est exécutée si la mémoire tampon de sortie 1 contient déjà des données qui n'ont pas encore été émises vers le maître SARTONET (SARTONET-master). C'est pourquoi les données de la mémoire tampon de sortie 2 sont émises en premier. Voici l'exemple d'une fonction qui écrit une chaîne de caractères dans la mémoire tampon 2:

```
struct s_write_2nd_buff {
            unsigned int        func;
            char                data[20];
    } buff2;

int writestring_buffer2(int v24_handle,
char *output)
    {
    int status;
    int len=strlen(output);

    buff2.func = WRITE_BUFFER2 ;
    strcpy(buff2.data,output);
    status = ioctl(fh,IOCTL_WRITE,&buff2,
    len+2 );
    return status;
    }
```

La commande IOCTL WRITE retourne «0» si la mémoire tampon 2 est déjà en train d'être utilisée.

Fonction 9: Change UART (redéfinir UART)

Cette fonction n'est applicable que sous le système d'exploitation MS-DOS.
Elle permet une modification de l'adresse UART sans modifier la ligne «device=....» dans le fichier CONFIG.SYS.

```
typedef struct s_change_uart {
        unsigned int            func;
        unsigned int            port;
        unsigned char           int_number;
    }   t_change_uart;

t_change_uart change_uart;

void main(void)
    {
    fh = open( V24 , O_RDWR | _BINARY);
         .
         .
    change_uart.func = CHANGE_UART;
    change_uart.port = 0x2f8;       /* COM2 port
                                       adress */
    change_uart.int_number = 0xb; /* COM2
                                     interrupt
                                     number */
    ioctl(fh,IOCTL_WRITE, &change_uart,
    sizeof(change_uart));
    }
```

Fonction 10: Set MUX Channel (définir le canal multiplexeur)

Cette fonction ne s'applique qu'aux interfaces MUX «M1 à M3» du YDI150 (adresse 0x300, interruption 0xF). Elle est utilisée pour commuter entre les trois canaux de l'UART.

Exemple:

```
/* definition area of the program.*/
...
...
...
struct { int set_chan_func;
       int muxport;
       }set_chan;
...
...
...
/* code area of the program */
...
...
...
set_chan.set_chan_func=10; /* function - no */
set_chan.muxport=port; /* allowed: 1...3 */
ioctl ( handle,IOCTL_WRITE,&set_chan,sizeof
(set_chan));
...
...
...
```

Commande I/O control READ

La commande IOCTL READ permet de lire l'état actuel du gestionnaire SER150. Vous obtenez toutes les informations à partir du format de langage C «io_ctrl» telle qu'elle est décrite aux paragraphes concernant les fonction MOUNT buffer page 98 et SET PROTOCOL page 99. Vous pouvez lire les informations dans leur totalité ou seulement une partie. La quasi-totalité des informations est normalement donnée par votre programme d'application. C'est pourquoi il n'est en fait pas nécessaire de toujours les relire en entier exepté les deux premiers octets dans le champ «dev_status» qui indiquent l'état actuel du gestionnaire de périphérique .

Le premier octet dans le champ «dev_status» est remis sur «0» après chaque instruction IOCTL READ et il donne l'état d'erreur. Le deuxième octet est lu lors de chaque instruction IOCTL READ et contient des informations générales sur l'état:

dev_status 2-Byte integer

- Bit 0: Overrun error (erreur de dépassement)
- Bit 1: Parity error (erreur de parité)
- Bit 2: Framing error (erreur d'encadrement)
- Bit 3: Buffer Overflow error (erreur de dépassement du tampon)
- Bit 4: Send Timeout (timea) (dépassement de temps pour l'émission)
- Bit 5: Receive Timeout (timeb) (dépassement de temps pour la réception)
- Bit 6: Wait Timeout (timec) (dépassement de temps pour l'attente)
- Bit 7: Block check error (modes SARTONET) (erreur de contrôle par bloc)
- Bit 8: Output buffer #2 in use (tampon de sortie 2 actif)
- Bit 9: Output buffer #2 in use (tampon de sortie 2 actif)
- Bit 10: Data available in current (unterminated) input block (données disponibles dans le bloc d'entrée en cours non terminé)
- Bit 11: Terminated data block available in input buffer (bloc de données terminé disponible dans le tampon d'entrée)
- Bit 12: Status of Clear To Send (CTS) line (état de la ligne CTS)
- Bit 13: Status of Data Set Ready (DSR) line (état de la ligne DSR)
- Bit 14: Status of Ring Indicator (RI) line (état de la ligne RI)
- Bit 15: Status of Data Carrier Detect (DCD) line (état de la ligne DCD)

L'exemple de programme suivant illustre deux fonctions qui utilisent IOCTL READ. La première fonction lit la totalité des réglages de protocole, la deuxième retourne l'état d'erreur (0, si aucune erreur n'est survenue).

```
read_protocol_settings(int v24_handle,
          struct io_control *io_ctrl)
    {
    ioctl(v24_handle,IOCTL_READ,
          io_ctrl,sizeof(struct io_control));
    }

int read_status(int v24_handle)
    {
    char status;
    ioctl(v24_handle,IOCTL_READ,&status,1);
    return status;
    }
```

Programme «V24_SET»

Le programme V24_SET.EXE peut être utilisé pour fixer les paramètres de transmission du gestionnaire de périphérique avec une entrée «DOS-Command Line» (par exemple dans un fichier batch), sans être obligé d'appeler la fonction IOCTL dans le cadre de votre application. Trois types de réglages sont possibles:

Réglages SARTONET MASTER

Pour configurer l'interface en tant que SARTONET-Master, il vous faut entrer l'instruction suivante:

```
V24_SET name SNET_MASTER [OBUF=len]
```

name Nom du gestionnaire de périphérique MS-DOS

len Longueur du tampon d'entrée et de sortie SARTONET (réglage par défaut = 256 octets)

Réglages SARTONET SLAVE

Pour configurer l'interface en tant que SARTONET-Slave, il faut entrer l'instruction suivante:

```
V24_SET name SNET_SLAVE=addr [tc]
```

name Nom du gestionnaire de périphérique MS-DOS

addr Adresse de l'esclave SARTONET (0..31)

tc Dépassement de temps en milli-secondes («timec») à la réception

Autres réglages de protocole

Pour exploiter l'interface avec un autre mode de protocole, il faut entrer l'instruction suivante:

```
V24_SET name baud parity wordlen stopbit ta tb
tc mode [ctl]
```

name	Nom du gestionnaire MS-DOS
baud	Vitesse de transmission (110,150,300,600,1200,2400,4800,9600, 19200,38400 bauds)
parity	Parité, O = Odd (impair), E = Even (pair), Z = Zero (zéro), N = None (pas de parité)
wordlen	Longueur du mot (5..8 bits)
stopbit	Nombre de stopbits (1..2 bits)
ta	Dépassement de temps en milli-secondes à l'émission («timer a»)
tb	Dépassement de temps en milli-secondes jusqu'à la réception du caractère de fin de bloc («timer b«)
tc	Dépassement de temps en milli-secondes jusqu'à la réception du caractère de début de bloc («timer c«)
	mode — mode de bloc-transfert:
	NONE — pas de transfert de bloc
	CHAR=ccc — ccc est le caractère de fin de bloc
	TIMEOUT — Interruption du transfert de bloc lors d'un dépassement du temps prévu «tb» entre deux caractères
	LEN=len — Longueur du bloc égale à la valeur prédéfinie «len».
ctl	Paramètres de contrôle: RI, CTS, DSR, CD, RTS, DTR activent les procédures de contrôle et de commande des lignes handshake XON=ccc, XOFF=ccc activent le protocole XON/XOFF pour le software-handshake OBUF=len fixe la longueur d'un tampon de sortie égale à «len»

L'exemple suivant active le pilote «ser2» avec 9600 bauds, hardware-handshake et transfert de bloc avec caractère de fin de bloc LF (0Ahex):

```
V24_SET SER2 9600 E 7 1 800 1000 1000 CHAR=/0x0a
DTR RTS CTS
```

Licence d'exploitation du logiciel Sartorius

Le présent document constitue un acte juridique et un accord entre vous, l'utilisateur ultime, et la société Sartorius.

En ouvrant l'enveloppe scellée contenant la disquette, ou en signant la fiche d'enregistrement, vous vous engagez à respecter les termes de cet accord.

Nous vous prions de ce fait de bien vouloir lire cet acte attentivement et dans sa totalité.

En cas de désaccord avec les termes du présent document, ne pas ouvrir l'emballage contenant la disquette, mais le retourner immédiatement où il a été acheté, accompagné de tout autre article (y compris documentation écrite, classeur ou autre reliures ou dossiers, et matériel, le cas échéant) faisant partie du produit pour remboursement intégral.

Termes du contrat

1. Objet du contrat

   Font l'objet de ce contrat la disquette contenant le programme logiciel, le manuel d'utilisation ainsi que toute la documentation écrite faisant partie de la livraison (le tout ci-après désigné «logiciel»).
   Sartorius tient à faire remarquer qu'au stade actuel de la technique, il n'est pas possible de créer des logiciels fonctionnant parfaitement pour toutes les applications et combinations. C'est la raison pour laquelle l'objet du présent accord est un logiciel exploitable par principe dans les limites fixées par les instructions d'utilisation qui l'accompagnent.

2. Octroi de licence

   Pendant la durée du contrat, Sartorius, le concédant, accorde à vous, le détenteur, le droit non-exclusif d'utiliser et d'afficher cet exemplaire d'un programme logiciel Sartorius sur un mono-ordinateur (avec une seule unité centrale) en un seul lieu, à condition que les termes de la licence soient respectés. Si le mono-ordinateur sur lequel est exploité le logiciel est un système à plusieurs utilisateurs, veuillez vous adressez à votre revendeur Sartorius pour obtenir une licence applicable à plusieurs utilisateurs. Le détenteur de la licence est autorisé à transférer physiquement le logiciel (c'est-à-dire mémorisé sur un support de données) d'un ordinateur à un autre, à condition toutefois que ledit logiciel ne soit exploité que sur un seul ordinateur à la fois. Toute autre exploitation du logiciel est strictement interdite.

3. Restrictions particulières

   La licence d'exploitation de ce logiciel et de la documentation qui l'accompagne n'est conférée qu'au détenteur. Celui-ci est donc tenu de respecter les interdictions suivantes:

a) Il est strictement interdit de transmettre le logiciel et la documentation qui l'accompagne à un tiers ou de le mettre à la disposition d'un tiers sous quelque forme que ce soit sans le consentement écrit préalable de Sartorius.
b) Les transferts électroniques du logiciel d'un ordinateur à un autre sur un réseau ou par l'intermédiaire d'un canal de transmission des données sont également strictement interdits sans le consentement écrit préalable de Sartorius.
c) Toute modification, «rétrotechnique», ou désassemblage du logiciel,
d) de même que toute création d'ouvrages tirés de celui-ci, toute reproduction de la documentation écrite l'accompagnant,
e) y compris toute modification ou rédaction d'ouvrages s'inspirant de cette même documentation est strictement interdite sans le consentement écrit préalable de Sartorius.

4. Droit de propriété du logiciel
Le détenteur de la licence est le propriétaire du support magnétique sur lequel est enregistré le logiciel Sartorius. Sartorius conserve cependant de façon intégrale le droit de propriété du logiciel enregistré sur la disquette ainsi que sur toute autre copie qui en aura été faite. En particulier, Sartorius se réserve les droits de publication, de reproduction, de traitement et d'exploitation du logiciel.

5. Reproduction
Le logiciel et la documentation écrite qui l'accompagne sont protégés par les droits d'auteur. Le détenteur est autorisé à faire une copie de secours et une seule du logiciel si ledit logiciel n'est pas protégé contre les reproductions. Il est strictement interdit de retirer une empreinte de copyright de même que les numéros d'enregistrement figurant sur celle-ci.
Toute reproduction non autorisée du logiciel, y compris sa modification, sa fusion ou son inclusion dans un autre, ou de la documentation écrite qui l'accompagne est expressément interdite.

6. Transfert du droit d'exploitation
La licence d'exploitation de ce logiciel ne peut être transférée à un tiers qu'avec le consentement écrit préalable de Sartorius et le bénéficiaire d'un transfert autorisé par Sartorius sera tenu aux termes et conditions de la présente licence. Il est expressément interdit de céder, louer ou prêter le logiciel.

7. Durée du contrat
La durée du contrat est illimitée.
La présente licence sera annulée automatiquement sans préavis par Sartorius au cas où le détenteur ne se conforme pas aux termes de celle-ci. En cas de résiliation, le détenteur devra immédiatement détruire toute documentation et tous exemplaires du logiciel, y compris les copies modifiées, le cas échéant.

8. Indemnité de violation du contrat

   Sartorius attire l'attention sur le fait que le détenteur répond de tout dommage subi par Sartorius suite à une atteinte aux droits d'auteur résultant du non-respect par le détenteur des termes de ce contrat.

9. Modifications et mises à jour

   La société Sartorius peut diffuser des versions révisées du logiciel qu'elle pourra, à son gré, mettre à la disposition du détenteur.

10. Garantie et responsabilité de Sartorius

    a) Sartorius garantit au détenteur initial de la licence qu'à la date de la livraison, la disquette sur laquelle est enregistrée le logiciel ainsi que le matériel qui accompagne éventuellement celle-ci sont exempts de tout défaut de matière première et de vice de fabrication dans des conditions normales d'utilisation et d'entretien.

    b) En cas de défectuosité de la disquette ou du matériel qui l'accompagne, le détenteur peut, pendant la période de garantie, exiger une livraison de remplacement. Il doit pour cela retourner la disquette, le matériel qui l'accompagne éventuellement, la copie de secours, la documentation ainsi qu'une copie de la facture au revendeur Sartorius qui lui a vendu le produit.

    c) Pour les raisons mentionnées au paragraphe 1 du présent acte, Sartorius n'assume aucune responsabilité en ce qui concerne l'exactitude et la fiabilité du logiciel. En particulier, Sartorius ne donne aucune garantie concernant l'aptitude du logiciel à satisfaire aux fins et exigences du détenteur ou la compatibilité du logiciel avec les programmes choisis par le détenteur. Tous risques afférents au choix, conséquences d'emploi, résultats et performances du logiciel sont supportés par le détenteur. Ceci vaut également pour la documentation qui accompagne le logiciel.

    d) Sartorius ne pourra être tenu responsable des dommages découlant de l'utilisation de ce produit sauf si un fait intentionnel ou une négligence grossière de Sartorius s'en avérait à l'origine. Il n'est pas dérogé à la garantie de la qualité éventuellement promise par Sartorius. Toute responsabilité pour des dommages résultant de vices non couverts par la garantie est exclue.

Informations juridiques

Remarques importantes destinées à l'acheteur et à l'utilisateur du programme logiciel ci-joint.

L'acquéreur/acheteur du programme logiciel ci-joint reconnaît que ledit logiciel est soumis aux droits de propriété, d'auteur et de reproduction ainsi qu'aux autre droits de protection. A aucun moment, l'acquéreur/acheteur n'acquiert un des droits susmentionnés.

Toute reproduction ou copie du logiciel est interdite sauf pour constituer une disquette de sauvegarde. L'acquéreur/acheteur n'est autorisé à utiliser le logiciel qu'à ses propres fins et ne pourra en aucun cas le céder à un tiers, que ce soit à titre gratuit ou à titre onéreux. En utilisant et en exploitant le programme, l'acquéreur/acheteur accepte les termes de la licence d'exploitation SARTORIUS jointe au programme original.

Sartorius AG
✉ 37070 Goettingen, Germany
📦 Weender Landstrasse 94–108, 37075 Goettingen, Germany
☎ (+49/551) 308-0, FAX (+49/551) 308-32 89
Internet: http://www.sartorius.com

Copyright by Sartorius AG, Goettingen, Germany.
All rights reserved. No part of this publication
may be reprinted or translated in any form or by any means
without the prior written permission of Sartorius AG.

The status of the information, specifications and
illustrations in this manual is indicated by the date
given below. Sartorius AG reserves the right to
make changes to the technology, features,
specifications and design of the equipment
without notice.

Status: April 1993, Sartorius AG, Goettingen, Germany

Printed in Germany on paper that has been bleached without any use of chlorine · W1A130 · KT
Publication No.: WYD6046-i93041

sartorius